REMARKS:

General:

By the above amendment, Applicant have amended the drawings for compliance with 37 CFR 1.84(p)(5) and 37 CFR 1.121(d), as requested. Applicant has also rewritten all claims to define the invention particularly and distinctly so as to overcome the technical rejections and define the invention patentably over the prior art.

Working and commercial model:

Please find enclosed a brochure titled "Player GX Technical Overview". We have a working and commercial model of the invention. It has so far been installed by 3,549 end-users, and demonstrations are available on our Web site.

Please also find enclosed a scientific paper titled "Compact multimedia format for digital video web interfaces". It contains information on some of the benefits of the format.

Regarding claim construction - MEP § 2106(II) and §2111.04:

Claims 1, 5, 7, 14 and 15 were commented to as not imposing any particular functional requirement, and therefore do not limit the claim. We have rewritten the claims to also be more functional and limiting of the claims. Also, the program instruction code have been described in better detail, as to what it contains ("Java byte code, assembler, or other code"), user input data handling (such as mouse, keyboard, gamepad, joystick, etc.), loading and rendering 3D models, and the program instruction code interface from the renderer ("API containing at least the classes and functions to get and set the attributes of a data blocks section").

Regarding claim objections - 37 CFR 1.75(c):

Claims 2-4 and 8-9 were objected to as being of improper dependent form for failing to further limit the subject matter of a previous claim. We have rewritten all claims and hope they now are proper and satisfactory.

Regarding claim rejections - 35 USC § 112:

Claim 3 was rejected for being indefinite. We have rewritten all claims and hope they now are definite and distinct. We have made sure there are no more "e.g." in the claims.

Regarding claim rejections - 35 USC § 101:

Claim 1 was rejected because the claimed invention lacks patentable utility. Claims 11-13 were rejected because the claimed invention is directed to non-statutory subject matter. We have rewritten all claims and hope they now have more clear patentable utility. We have also made sure they are not claiming any invention directed to non-statutory subject matters.

Regarding claim rejections - 35 USC § 102:

Claims 1-3, 5, 7-8, 10-12, and 14-15 were rejected under 35 U.S.C. 102(b) as being anticipated by Goetz et al. We have rewritten all claims to make sure they satisfy 35 U.S.C 102.

Goetz et al. primarily teaches a file format and system for streaming video and other media over a TCP or UDP network. Goetz anticipates a relatively typical streaming system as illustrated in their figure 10, where their web client application cooperates with their web server application to produce their multimedia file from the server. Goetz describe the interaction and cooperation (primarily in detailed description "a. Set Up" and "b. Streaming"). Goetz' invention describes a typical streaming system, like e.g. Microsoft Windows Media Player and Microsoft Windows Media Server. The setup process is similar. Further, Goetz' primarily teaches (column 12-16) how to stream video and media over a UDP network, dealing with well-known UDP issues, such as bit rate throughput, network jitter, round-trip delay and percentage and distribution of packet loss. The file format, teached by Goetz, is similar many other streaming formats, and primarily deals with packetizing video and media for UDP transfer.

As we mention in the background section of our specification, we were quite aware of these types of streaming formats that Goetz teaches. Our invention was made to overcome the technical limitations of these formats. As presented in the enclosed scientific paper "Compact

multimedia format for digital video web interfaces" and the specification, our invention has cardinal improvements. A computer game, for example, can contain hundreds of gigabyte of 3D models, textures, program code, videos, and other media, stored in tens of thousands of files. Without the benefits discussed in our paper (analysis), and our invention, the format and system would not work.

Goetz et al did not, by far, anticipate our invention. It is not possible to adapt Goetz format ("Directory Preamble 410" figure 4B on column 7 lines 65-68") or ("Packet Descriptor 420" figure 4C and column 8 lines 5-6) to store a scene block header, or anything else, because it is designed for (network) packet descriptions. Packetizing is related to fitting media data into UDP network packets (that must be smaller than 10 KB on most networks), and dealing with all the typical technical problems of UDP, such as packetloss, latency, late-delivery, etc. Neither is it possible to use ("Media Block Body 430" figure 4D), ("Packets 440" figure 4D) or ("media block delivery 320"). It's not designed for that. "Start time 749" and "end time 750" cannot replace the program instruction code.

Regarding claim rejections - 35 USC § 103:

Claims 4, 9, and 13 were rejected as unpatentable over Goetz et al in view of Wan (International Publication No. 02/05089 A1). We have rewritten all claims to make sure they satisfy 35 U.S.C 103.

Applicant submits that the rejections would be improper because the combination of Goetz and Wan (and the other references) fails to teach or suggest each of the claim limitations. For claim 1, for example, the combination fails to teach at least "the software renderer (103) sends a request packet to a source computer (100) containing the logical name of a 3D model and subset selection criteria consisting of specific keyframe animations, polygon reduction, selecting polygons within a 3D volume space, or selecting textures and media".

Conclusion:

We respectfully request that the new claims be considered. Examiner is invited to contact Applicant at the below-listed e-mail address, or telephone number, if it is believed that the prosecution of this application may be assisted thereby. Although only certain arguments regarding patentability are set forth herein, there may be other arguments and reasons why the claimed invention is patentable. Applicant respectfully reserves the right to raise these arguments in the future.

Request for Constructive Assistance: Applicant has amended the drawings and claims of this application so that they are proper, definite, and define novel structure, which is also unobvious. If, for any reason this application is not believed to be in full condition for allowance, applicant respectfully requests the constructive assistance and suggestions of the Examiner pursuant to M.P.E.P. § 2173.02 and § 707.07(j) in order that the undersigned can place this application in allowable condition as soon as possible and without the need for further proceedings.

Dated: November 12, 2008

Very respectfully,

By _____

Ole-Ivar Holthe
PhD candidate
Gridmedia Technologies AS
268 Bush Street #4229
San Francisco, CA 94104
(415) 283 9182 cell
(415) 567 7157 office and fax
ole@gridmedia.com

Certificate of Mailing: I hereby certify that this correspondence, and attachments, will be deposited with the United States Postal Service by First Class Mail, postage prepaid, in an envelope addressed to "Box Stop Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450" on the date below.
Date: 2008 nov. 12 Signature: _____.

Player GX™

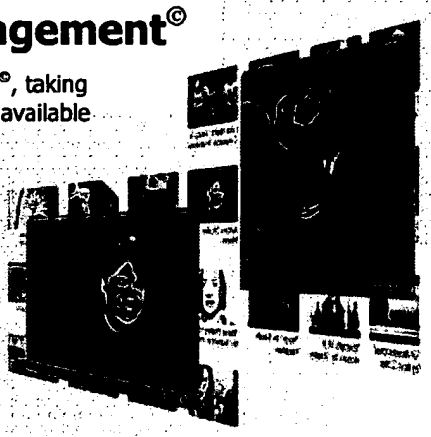
Technical Overview

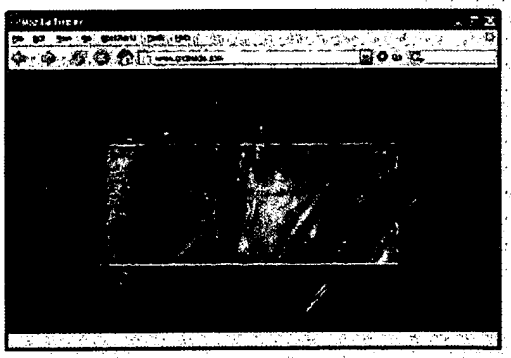
Player GX™ is a web browser plug-in software component. Extending the latest in graphics processing and streaming video technology, Player GX provides the ultimate in high quality, cutting edge, streaming video and rich media playback experiences on the Web.

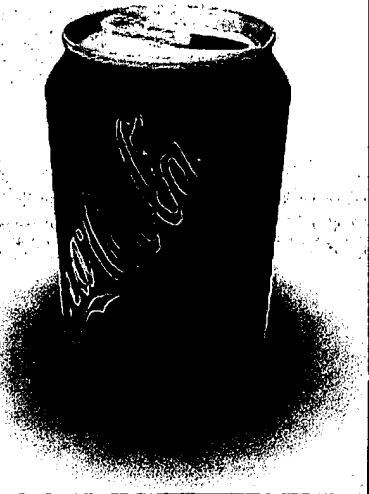
Compose	GXML, XAML, SVG, X
Graphics	DirectX 9.0/8.0, OpenGL 2.0, GDI/GDI+, Quartz, Shader Model 3.0/2.0/1.0
Codecs	WMV, MPEG, MOV, RM, +custom codecs
Language	Java 2 (J2ME VM included)
Browsers	IE, Netscape, Firefox, Mozilla, Opera
Platforms	Win32, Win64, Mac OSX, Linux

Advanced Presentation Management©

Player GX features Advanced Presentation Management©, taking full advantage of graphics hardware acceleration features available on the end-users computer. Scenes are composed of images, text, vector graphics, video and 3D elements, grouped and layered with animations, and fully interactive and programmable.







Real-time rendering of high quality interactive 3D graphics is used to create impressive menus, visualizing products, visualizing data, and providing advanced effects for streaming video.

Player GX provides integrated support for popular 3D tools, such as 3D Studio™, Alias|Wavefront™, and Maya™. The DirectX X-Format is fully supported, including associated textures, effects and shaders. The models are fully interactive and programmable.

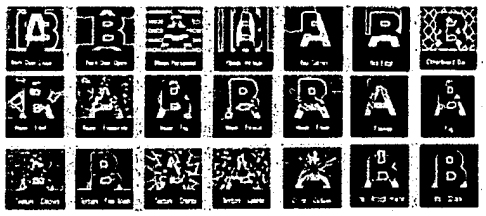
Email: gridmedia@gridmedia.com
Web: www.gridmedia.com

US Office (San Francisco):
tel: (415) 283 9182

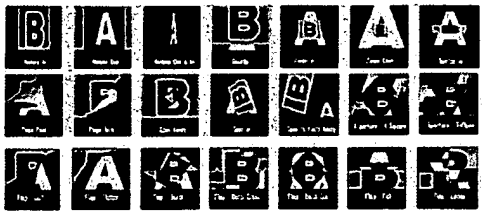
Europe Office (Norway):
tel: +47 90935742

Advanced Effect System©

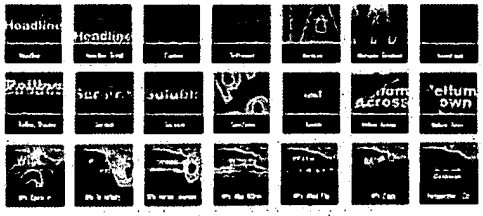
Creative use of audiovisual effects is essential for achieving great user experiences. The Advanced Effect System© provides pixel-, vertex- and text- effects, in addition to DirectShow and DMO filters. Below are a few select transition effects:



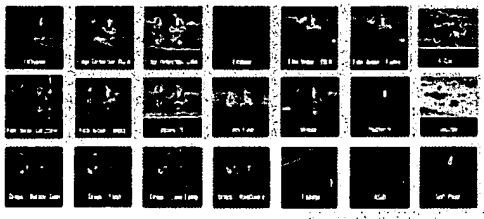
Pixel Effects



Vertex Effects

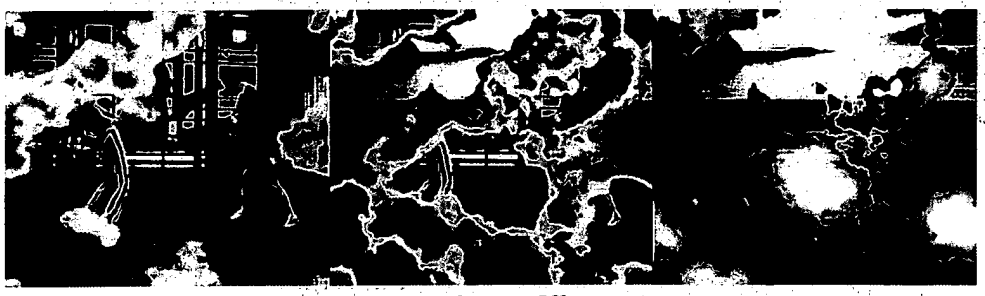


Text Effects



DMO Filters

Hardware Accelerated Effects provide the ultimate in high-quality post processing of streaming video and graphics. Hardware Accelerated Effects are supported with the High Level Shader Language (HLSL), the Shader Assembler, Effect Files, and the GX API. Intelligent Scalable Content© (next page) ensures graceful degradation for computers with lesser graphics capabilities.



Example Burn Effect

Example Ripple Effect

High-Level Shader Language (HLSL) is a C like programming langue for writing Shaders for GPU's. The compiled binary Shader instructions are uploaded to the GPU for real time processing. There are essentially two types of Shaders; Vertex- and Pixel- Shaders. Vertex Shaders perform programmatic processing of vertexes, while Pixel Shader perform programmatic processing of textures. Shaders are processed in real time by the GPU.

Email: gridmedia@gridmedia.com
Web: www.gridmedia.com

US Office (San Francisco):
tel: (415) 283 9182

Europe Office (Norway):
tel: +47 90935742

Unleashing Streaming Video

Seamless browsing in and between video streams, eliminating buffering delays, is paramount in providing the ultimate user experience. In addition to supporting video and audio codecs provided by DirectX, and custom DirectShow and DMO codecs, Player GX provides tight integration with the popular Windows Media™ 9-Series format and Windows Media™ Server.

The GX API provides direct access to important Windows Media functionality, such as;

- opening new streams, prerolling, start, speed, pause, resume, stop, start at marker,
- synchronization with animations and other streams,
- local caching, buffer management, monitoring buffers and caches,
- reading headers (bitrate, author, title, markers),
- managing output formats, managing stream selection (multibitrate, multistream), custom stream formats (scripts),
- DRM individualization, DRM license acquisition, and storing DRM licenses.

Video frames and audio signals are processed by elements and surfaces in the scene composition, and rendered by the Advanced Presentation Management© system.

Digital Rights Management

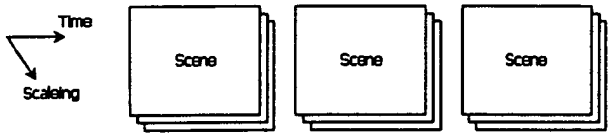
Windows Media™ Digital Rights Management (DRM) enables the content provider to secure Windows Media files from piracy, and other forms of illegal use. Windows Media DRM is currently the only DRM-technology accepted by the motion pictures industry.

Player GX provides essential enhancements to the user experience of using Windows Media DRM protected content. The GX API provides functionality for acquiring and storing licenses programmatically, avoiding the need for special predelivery steps or license acquisition dialog boxes.

Intelligent Scalable Content©

There exists a large variety of computers with varying graphics card capabilities, CPU, memory and network connections. Intelligent Scalable Content is an essential feature of Player GX, making sure that the content on the Web Site is available to everybody, no matter what kind of computer or bandwidth the user may have.

Player GX detects the capabilities of the users computer and network characteristics, which can be used for automatic selection of which scene to play, or manual actions using the GX API. Typically, one should provide a specific scene for users with high-class computers, providing broadband content and hardware accelerated graphics, and a specific scene for users with low class computers.



Additionally, Player GX supports advanced custom scalable content, with the feature rich GX API, for Web sites that require very specific adaptation of the content. GX API includes functionality for detailed inspection the capabilities of the graphics card, inspecting the CPU, memory, operating system, and different schemes for detecting network characteristics.

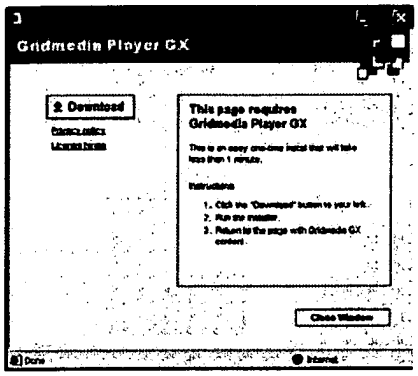
Email: gridmedia@gridmedia.com
Web: www.gridmedia.com

US Office (San Francisco):
tel: (415) 283 9182

Europe Office (Norway):
tel: +47 90935742

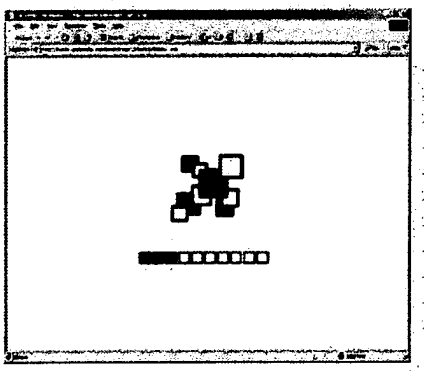
Fast and Easy Setup

The installation procedure for Player GX (450 KB) will take less than 1 minute, even for users with slow dialup connections.



Typically, a special Web page or popup window, as seen here, is provided to assist the user in installing the plug-in. The visual design of the web page (HTML) is easy to modify to accommodate specific design guidelines.

If DirectX or Windows Media are not already installed on the users computer, Player GX will detect this and may optionally install the missing components.



The visual design of the installation scene is easy to modify to accommodate specific design guidelines.

System Requirements

Supported operating systems:

- Microsoft Windows 95/2000/XP or later versions
- Microsoft Windows XP Media Center Edition 2005 and later versions
- Macintosh OS X (coming soon)
- Linux (coming soon)

Supported web browsers:

- Internet Explorer
- Netscape Navigator
- Opera
- Mozilla
- Firefox

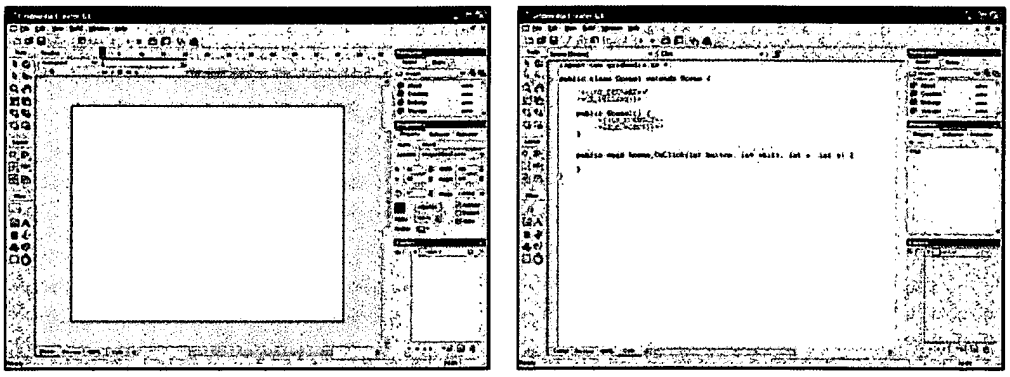
Email: gridmedia@gridmedia.com
Web: www.gridmedia.com

US Office (San Francisco):
tel: (415) 283 9182

Europe Office (Norway):
tel: +47 90935742

Creator GX™

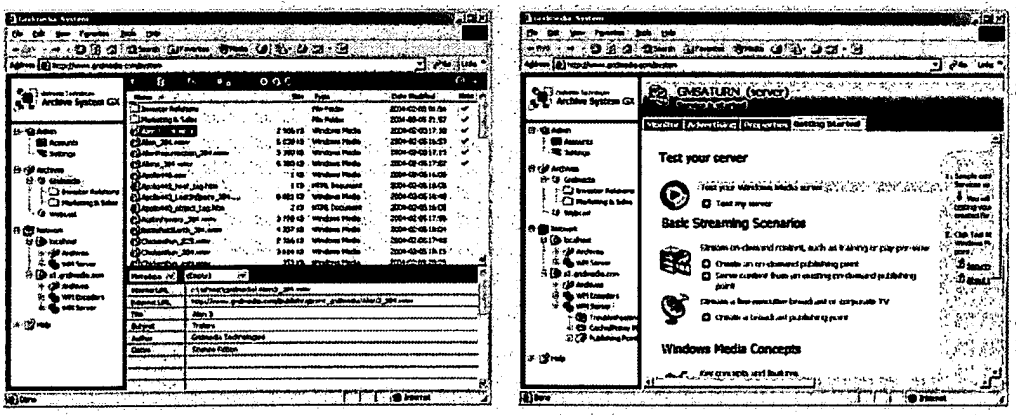
Creator GX™ is the content creation tool, used by web developers for creating GX formatted content (GXML).



Creator GX features an easy and intuitive What You See Is What You Get (WYSIWYG) user interface, using well-known concepts that are familiar to web designers, such as layers, groups, elements, animations, timers, etc. Elements, such as images, text, video, etc. are dragged on to the scene. Advanced Dynamic Layout© provides advanced dynamic layout behavior when the scene is resized to the containing web browser. The scene may be programmed with the Java-based programming language, providing access to the feature rich GX API.

Archive System GX™

Archive System GX™ is the server-side Multimedia Control Panel© system used for simplifying the creation and management of rich media Web sites.



Archive System GX is integrated with Creator GX and Player GX, to provide web developers with an easy and intuitive control panel for managing the Web interface, with respect to both archived and live content:

- Video and audio formats: Windows Media Format 7/9+ (WMF), QuickTime, Real, and MPEG-1/2/4/+
- Asset formats: GXML, GIF, GGF, JPEG, PNG, HTML, GX-HTML, OBJ (3D), G3D, SVG, GVG, X, FX, ...

Archive System GX provides integrated support for managing Windows Media Servers, making it easy to set up advanced streaming scenarios, featuring both live and on-demand content.

Email: gridmedia@gridmedia.com
Web: www.gridmedia.com

US Office (San Francisco):
tel: (415) 283 9182

Europe Office (Norway):
tel: +47 90935742

COMPACT MULTIMEDIA FORMAT FOR DIGITAL VIDEO WEB INTERFACES

Ole-Ivar Holthe[1] and Leif Arne Rønningen[2]

[1]Gridmedia Technologies AS,
N-7318 Agdenes, Norway
[2]Norges Teknisk Naturvitenskapelige Universitet,
N-7034 Trondheim, Norway

ABSTRACT

The next generation of web interfaces are emerging with the global adoption of broadband. Powered by complex server-side database-, application-, and media-servers, the next generation web interfaces will feature vast multitudes of interactive services integrated with interactive streaming video and rich media, scaled to fit the demands of the end-user and service providers. The number of content items, such as images, video clips, etc., is expected to grow substantially. In this paper, we introduce a highly compact multimedia format for containing multimedia files. The format supports existing infrastructure, and enables easy containment of multimedia files with integrated support for managing loading order, linking of external references and scalable content.

INTRODUCTION

There exist quite a few formats for storing and transferring multimedia content on the Internet. Traditionally, these were binary formats that described, at the bit level, the purpose of each individual bit, typically starting with some kind of header, followed by the data. Popular binary image formats include JPG, PNG, and GIF, which are all binary formats. JPG and PNG are both international standards that are continuously evolving with new features. The difference in features has made both formats popular and widely adopted. GIF is a proprietary format that is able to hold simple animations of images making it popular for "livening up" web pages. Popular binary multimedia formats include the MPEG 1,2,4 formats [4], the QuickTime format (QT) [3], Microsoft Active Stream format (ASF) [8], and Macro-media's SWF format [2]. MPEG, QT, and ASF are "streaming" video formats, which means that they primarily contain video data that can be consumed simultaneously with being transferred. The Macromedia SWF format is a highly popular format, used for containing Flash interfaces with associated resources.

In the 90s, the HTML standard [7] became immensely popular for defining web interfaces. It's textual tag-based structure made it very flexible and easy to understand and author content for. The vague definition of the standard further allowed browser manufacturers to extend the format with their own proprietary features. Later developments have given us extensions, such as JavaScript/DOM, cascaded style sheets, DHTML, etc. The XML standard [1] provides a way to standardizing the syntax of formats. You can theoretically represent any data with XML. For relatively small amounts of data, XML is an ok format. But, for large amounts of binary data, such as audiovisual data, XML is no good. HTML solves this by not including audiovisual data, but rather linking to it using SRC-tags.

New multimedia format developments include the MPEG-7 [5] and MPEG-21 [6] formats. Both are XML-based formats for describing, defining and transcoding "digital items". The formats are designed to externally reference binary content (e.g. audiovisual content). The new MPEG formats include tools for compressing text-based XML document into binary form, based on the XML Schema [1].

Our experiences with developing the next generation of rich multimedia interfaces for the web show that the number of external content items (e.g. audiovisual content) is much higher than with traditional HTML file. Rich multimedia web interfaces contain animations, layers and multiple scenes depending on scalable content considerations, comprising a large number of content items. Although it is feasible to maintain many of these content items as externally referenced files, it is easier to manage and maintain most of these content items in "container files".

In this paper, we introduce a "compact multimedia format", called "GXF", for containing multimedia files. GXF is well suited for efficient use on any class of computer, from computers with very limited hardware resources (e.g. handheld devices like mobile phones, PDA's and set-top boxes for Interactive TV), to computers with powerful hardware resources. GXF uses a block-based format for holding and/or describing multimedia content. Since the block-based format is relatively flat and uncomplex, in its data structure organization, is

easy to process and render on the destination computer. This results in a very small renderer implementation, and very low use of hardware resources, on the destination computer.

GXF is flexible with respect to the different media types and/or program code types that it may contain. The block-based structure of the format makes it easy to extend with a vast variety of media types. Depending on the value of the type field, the header and data blocks may contain a large number of different media types, limited only by the different renderer implementations. GXF provides good support for content scaling. The author can scale the scene with respect to bitrate (bandwidth), language (Norwegian, English, etc.), screen (resolution, refresh rate, etc.), and machine (computer class). Furthermore the author may split the scaled content into multiple files that are linked together using an external_link field, which is important for rapid loading of a specific content scaling by the destination renderer.

GXF is very efficient with respect to compactness in holding multimedia content. The individual blocks, or data in the blocks, may use different compression schemes, such as ZLIB, PNG, or JPEG compression. The author may specify which compression scheme to use in the content creation process. GXF provides streaming transmission, so that the destination renderers can render the multimedia content while the multimedia content is being transmitted over the transport medium. GXF uses resources to store the different media types, which the scenes use. The resources may be stored in any order by the content creation process, which gives the content author the ability to specify in which order the resources should be loaded, when streamed over a transmission medium. This is very important on slow transport mediums.

FORMAT DESCRIPTION

FIG. 1 depicts the basic logical organization of GXF content. It is up to the author to fill in the contents of the GXF content in accordance with this format. The GXF content is divisible into a main header section (300), a block headers section (301) and a data blocks section (302). In general, the header sections (300 and 301) are first transmitted from the source computer to the destination computer so that the destination computer may process the information within the header section. Subse-quently, the data blocks section (302) is transmitted from the source computer to the destination computer on a block-by-block basis.

The main header section (300) as illustrated in FIG. 1 contains information about the GXF content. The signature (310) specifies the main type of the GXF content, and is typically a large number that is unique for a specific authoring environment. The byte_count (311) specifies the total number of bytes contained in the GXF content. The block_count (312) specifies the total number of blocks (external or internal) contained in, or used, by the GXF content. The major_version (313), minor_version (314), major_revision (315), and minor_revision (316) specify the version of the GXF content format. The ex-tra_data (317) provides extra information about the GXF content, depending on the specific implementation of the GXF format. The extra_data (317) is optional, and may consist of a variable number of bytes, depending on the specific implementation.

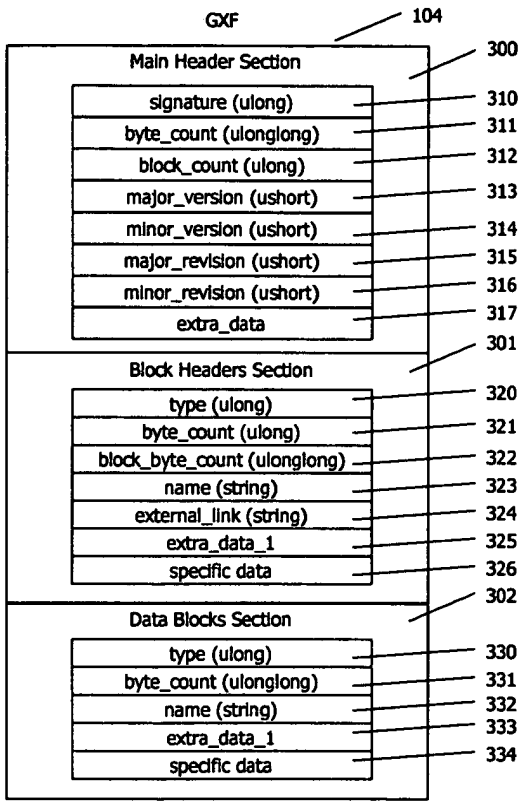


Fig. 1

The block headers sections (301) as illustrated in FIG. 1 contain a number of block headers that provide information about the GXF content. The number of block headers is specified by block_count (312) in the main header section (300). The information contained in a block header may vary, depending on the type of content that it describes. A block header will always begin with the fields as indicated in FIG. 1. The type (320) indicates the type of content that the header describes; this can for example indicate a scene, an image resource, or a text resource. The byte_count (321) specifies the total number of bytes in the block header. The block_byte_count (322) specifies the total number of bytes in the associated data block. The name (323) specifies the name of the content item. The external_link (324) specifies a link to the external GXF content, in which the associated data block is contained. The exter-

nal_link is empty if the associated data block is contained in the current GX content. The extra_data_1 (325) provides extra information about the block header and/or content item, depending on the specific implementation of the GXF format. The extra_data_1 (325) is optional, and may consist of a variable number of bytes, depending on the specific implementation. The specific data (326) may contain addi-tional information about the content item.

The data blocks section (302) as illustrated in FIG. 1 contain a number of data blocks that contain the data of the content items in the GXF content. The number of data blocks in the GXF content is equal to the number of block headers in the GXF content with an empty external_link. There exists exactly one data block for each block header with an empty external_link in the GXF content. The data blocks are specified in the same order, and are of the same content type, as the block headers. The type (330) indicates the type of content that the data block contains; this can for example indicate a scene, an image resource, or a text resource. The byte_count (331) specifies the total number of bytes in the data block. The name (332) specifies the name of the content item. The extra_data_1 (333) provides extra information about the data block and/or content item, depending on the specific implementation of the GXF format. The extra_data_1 (333) is optional, and may consist of a variable number of bytes, depending on the specific implementation. The specific data (334) may contain additional information about the content item.

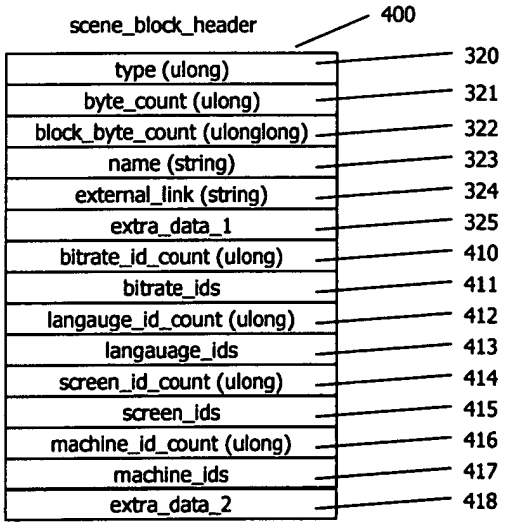


Fig. 2

The scene content type can be used in GXF content to represent the visual layout of multiple content items of different types. There can be multiple scenes in one GXF file. The scene can also be scaled (content scaling) by the renderers for different representations depending on the characteristics of the destination computer.

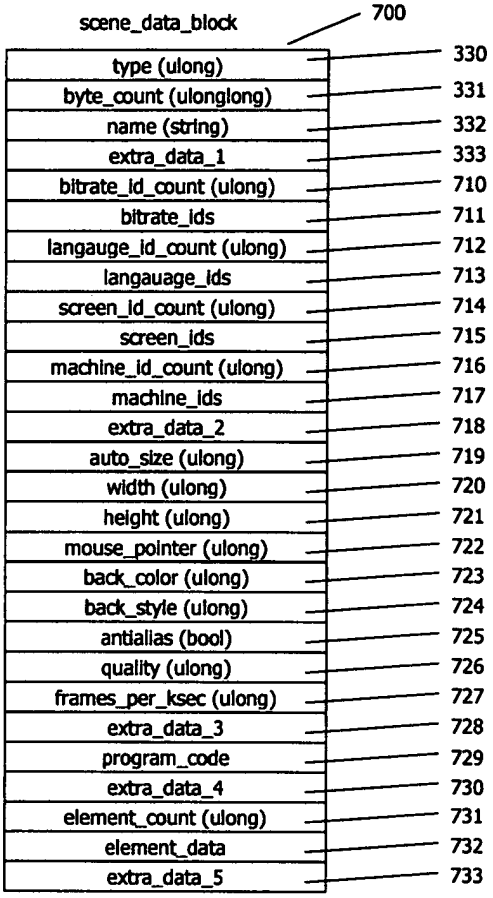


Fig. 3

The scene_block_header (400) as illustrated in FIG. 2 contains the block header data for the associated scene data block. The scene_data_block (700) as illustrated in FIG. 3 contains the scene data. The type (320 and 330) indicates that the type of the content item is of the scene content type. The bitrate_ids (411 and 711) specifies the bitrate identifiers used for content scaling. The bitrate_id_count (410 and 710) specifies the number of bitrate identifiers. The language_ids (413 and 713) specifies the language identifiers used for content scaling. The language_id_count (412 and 712) specifies the number of language identifiers. The screen_ids (415 and 715) specifies the screen identifiers used for content scaling. The screen_id_count (414 and 714) specifies the number of screen identifiers. The machine_ids (417 and 717) specifies the machine identifiers used for content scaling. The machine_id_count (416 and 716) specifies the number of machine identifiers. The bitrate_ids, language_ids, screen_ids, and machine_ids, may in an embodiment be of the unsigned long data type. The extra_data_2 (418 and 718) provides extra information about the scene block and/or content item, depending on the specific implementation of the GXF format. The ex-

tra_data_2 (418 and 718) is optional, and may consist of a variable number of bytes, depending on the specific implementation. The remaining properties are specific to the scene content type. Similarly, specific properties may be associated to images (type, width, height, bit count), and other content item types.

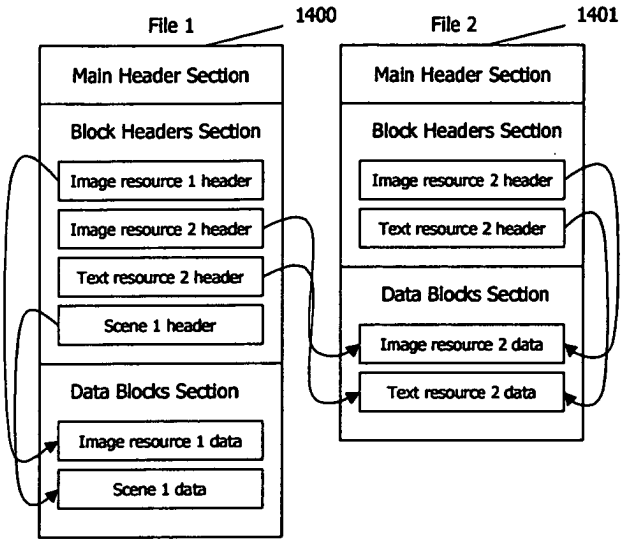


Fig. 4

FIG. 4 is an example on how content can be effectively linked together for the purpose of efficient transmission of the multimedia content over a slow transmission medium. Typically, the main problems with a slow transmission medium are; high access time and low transmission rate. The access time is the time from the destination computer requests content, until the destination computer initially receives it. The transmission rate is the rate at which data can be delivered across the transmission medium. The GXF format can embed many small content items as resources, which reduces the total content transfer time on transmission mediums with a high access time. As one can see in FIG. 4, the GXF files (1400 and 1401) contain multiple data blocks, which contain content items, in each GXF file. The arrows in FIG. 4 illustrates content linking, using the external_link (324) field of the block headers. The external_link field indicates where the data block is located, either in the same file, or an external file. The external_link field may be an URL. By linking multimedia content in this manner, one can have efficient reuse of multimedia content between different GXF files, while maintaining a minimal number of GXF content files. Reuse of multimedia content is important, since it can be used to avoid having to retransmit the same content item multiple times. You do want to avoid retransmission of content items on slow transmission mediums.

ANALYSIS

Here we consider two representations of a scene (S1 and S2), scaled for different classes of computing environments. Each scene uses one resource (R1 and R2).

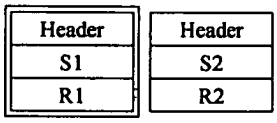


Fig. 5

FIG. 5 illustrates two GXF files, each containing a scene with the associated resource. The left file is the main/default file, containing header links to the right file. The playback mechanism will begin loading the header of the left (default) file first. Depending on the end user computer capabilities, the playback mechanism will either proceed to load S1+R1, or proceed to load S2+R2 from the right file.

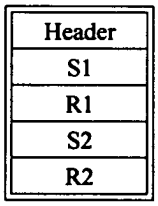


Fig. 6

FIG. 6 illustrates one GXF file, containing both scenes with the associated resources. In this approach, the playback mechanism will have to load S1+R1 prior to loading S2+R2.

We wish to determine the total access time to load S2 and R2, for both approaches (A and B), with varying network access time. We consider a network with transmission rate of 800 kbps, and assume the binary size of 1kB for each of the scenes and resources.

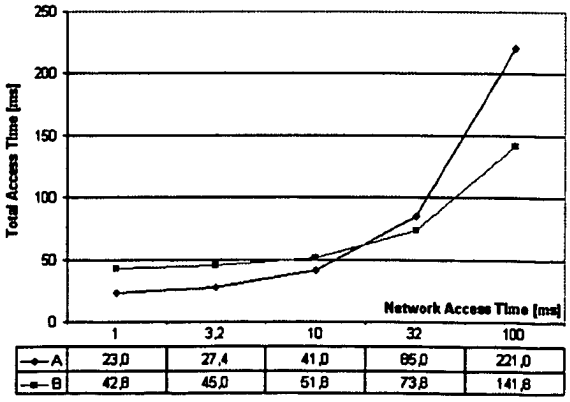


Fig. 7

FIG. 7 shows that approach A (FIG. 5) is more efficient for networks with low access times, while approach B (FIG. 6) is more efficient for networks with high access time.

For slower networks, the difference between the approaches becomes more pronounced. In FIG. 8, we consider a network of transmission rate 8 kbps and resource binary sizes of 256 bytes.

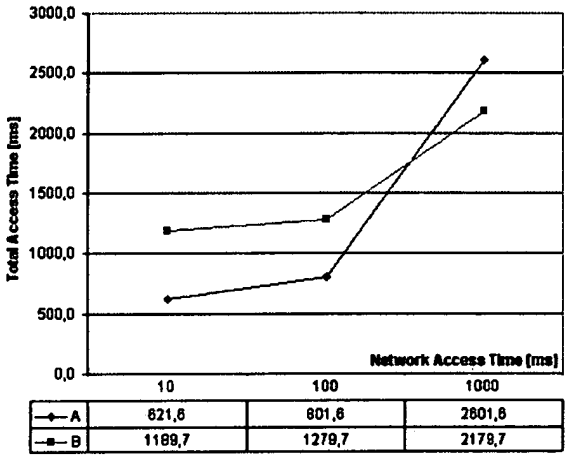


Fig. 8

Typically, you wish to provide scaled representations of scenes for:

- "High class" playback,
- "Low class" playback, and optionally
- "Medium class" playback.

The "high class" representation is designed for end-users having a powerful computing environment (CPU, graphics, network, etc.). Lower class representations are provided for weaker computing environments (hand-helds, etc.). You may also wish to provide different representations for different natural languages (English, German, etc.).

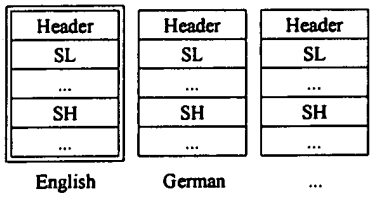


Fig. 9

FIG. 9 illustrates a possible "file layout scheme" for the low-class scene (SL) and the high-class scene (SH) for different natural languages. If the binary size of SL is sufficiently small, the high-class playback will not suffer much from the overhead, while preserving fast access for English language low-class playback.

DISCUSSION

For environments with high-class computers and high-speed networks, GXF provides little value beyond making it easier for web designers to group multimedia content items for manageability. The web today, however, is becoming increasingly complex. For slow devices and slow networks, complexity and manageability of transmission is essential for achieving good quality of service. GXF enables the web designer an easy and intuitive method for achieving scalability and control of the loading order.

An important problem with multimedia formats and external linking has to do with how web servers serve content. Typically used web servers, such as Apache and Microsoft Internet Information Server, parse the HTML file for externally referenced content files prior to sending the HTML content, and attaches the content files to the response. This avoids the problem of multiple accesses to the web server, which is necessary to achieve good performance when serving HTML pages. These web servers do not currently recognize new formats, such as MPEG-7 or MPEG-21. Embedding new formats and resources in the GXF file, or having a trailing GXF file on the XML document, will bypass this problem.

REFERENCES

[1] "An MPEG-7 Tool for Compression and Streaming of XML Data", U. Niedermeier, J. Heuer, A. Hutter, W. Stechele, A. Kaup, IEEE International Conference on Multimedia and Expo (ICME 2002), Lausanne, Switzerland, August 26-29, 2002.

[2] Macromedia SWF Format: http://www.openswf.org

[3] Apple QuickTime Format: http://www.apple.com

[4] ISO/IEC 14496: Coding of Audiovisual Objects: Systems. (MPEG-4).

[5] ISO/IEC JTC1/SC29/WG11 N4980: MPEG-7 Overview (version 8).

[6] ISO/IEC 21000-9: MPEG-21 File Format.

[7] The HTML standard: http://www.w3c.org

[8] Microsoft ASF Format: http://www.microsoft.com